

# Hierarchical Aggregations for High-Dimensional Multiplex Graph Embedding

Kamel Abdous, Nairouz Mrabah, Mohamed Bouguessa

**Abstract**—We investigate the problem of multiplex graph embedding, that is, graphs in which nodes interact through multiple types of relations (dimensions). In recent years, several methods have been developed to address this problem. However, the need for more effective and specialized approaches grows with the production of graph data with diverse characteristics. In particular, real-world multiplex graphs may exhibit a high number of dimensions, making it difficult to construct a single consensus representation. Furthermore, important information can be hidden in complex latent structures scattered in multiple dimensions. To address these issues, we propose HMGE, a novel embedding method based on hierarchical aggregation for high-dimensional multiplex graphs. Hierarchical aggregation consists in learning a hierarchical combination of the graph dimensions and refining the embeddings at each hierarchy level. Non-linear combinations are computed from previous ones, thus uncovering complex information and latent structures hidden in the multiplex graph dimensions. Moreover, we leverage mutual information maximization between local patches and global summaries to train the model without supervision. This allows to capture globally relevant information present in diverse locations of the graph. Detailed experiments on synthetic and real-world data illustrate the suitability of our approach on downstream supervised tasks, including link prediction and node classification.

**Index Terms**—Multiplex Graphs, Graph Representation Learning, Graph Neural Networks.

## 1 INTRODUCTION

### 1.1 Context and Background Information

TODAY'S real systems are mostly made of entities that interact with each other through multiple channels of connectivity. To better represent such complex systems, a new class of graphs, called *multiplex graphs*, has emerged. In a multiplex graph, nodes (entities) are connected to each other through multiple types of relations (links) [1]. We refer to these relations as dimensions. Such graphs arise in various areas, such as protein-protein interactions [2], neuroimaging [3], social networks [4], online recommendation [5]. In general, multiplexes provide a more general framework that allows rich and flexible modeling of several interconnected systems.

In recent years, many efforts have been made to design effective mining algorithms for multiplex graphs [6], [7], [8], [9], [10]. Most of these approaches are based on graph embedding techniques, which aim to project the graph dimensions to a compact and low-dimensional latent space representation. An effective embedding method should achieve exploitable representations for various prediction tasks, such as node classification [11], [12] and link prediction [13], [14]. This is not straightforward because multiplex graphs may have a large number of dimensions, each containing various complementary and / or divergent information on node interactions [7]. Therefore, an optimal embedding method should only encode the intrinsic informative structures hidden in the graph dimensions.

Over the past few years, a number of embedding methods for multiplex graphs have been proposed. Some of these

methods [14], [15], [16] generalize random walks [17] to the multiplex scenario. They either perform random walks on individual dimensions and then aggregate the results or run random walks that jump from one dimension to another. However, with high-dimensional multiplex graphs, long sequences of nodes are needed to account for all dimensions. Since these methods optimize the embeddings on local patches, it becomes difficult to extract global information. Another line of work uses consensus embeddings to encode information of multiple dimensions [18], [19], [20]. A consensus embedding is the representation of a node that includes all dimensions of the graph [18]. In particular, these methods leverage regularization to force the embeddings to be similar across dimensions. This strategy is effective in some cases, but in the presence of complementary dimensions, it can hinder the quality of the final representations.

The success of deep neural networks in encoding complex data is effectively aligned with learning appropriate embeddings for multiplex graphs [21]. In this regard, Graph Neural Networks (GNNs) [22], [23] extend the deep learning framework to graph-structured data. The most successful attempts to extract latent representations for multiplexes leverage GNNs [12], [21]. Most of these methods construct separate embeddings on individual dimensions with GNNs and then aggregate the dimension-specific embeddings into a consensus one using a linear weighted summation. However, these methods consider a single and simple combination. A linear aggregation of the dimension-specific embeddings cannot capture complex relations between the graph dimensions. Moreover, it is not clear from previous work how to exploit several aggregations probably because combined representations from different sets of dimensions lead to inconsistent results [24]. In this context, we argue the existence of relevant hidden dimensions, which can

- K. Abdous, N. Mrabah, M. Bouguessa are with the Department of Computer Science, University of Quebec at Montreal, Montreal, QC, Canada. E-mails: [abdous.kamel@courrier.uqam.ca](mailto:abdous.kamel@courrier.uqam.ca), [mrabah.nairouz@courrier.uqam.ca](mailto:mrabah.nairouz@courrier.uqam.ca), [bouguessa.mohamed@uqam.ca](mailto:bouguessa.mohamed@uqam.ca)

be established hierarchically from the dimension-specific embeddings. Thus, a high-level relation can be seen as a composition of lower-level ones. Unfortunately, existing aggregation strategies cannot handle this compositional aspect.

## 1.2 Motivations

Multiplex graphs are increasingly characterized by the presence of a high number of dimensions (that is, a large number of links of different types connecting multiple nodes). In this setting, informative and complex latent structures can be hidden across various dimensions. Mining such high-dimensional multiplex graphs continues to pose a challenge to existing methods. In particular, the number of possible combinations grows exponentially with the number of dimensions. Aggregating all dimensions in a *single linear* step can cause significant information loss. To explain these problems, we provide two illustrative examples.

**Single vs Multiple:** We consider the case of a co-authorship multiplex graph, where nodes are the authors of research papers, and edges indicate that two authors have co-written a paper. Edges can be scattered in multiple dimensions such that each dimension represents a publication venue (that is, two authors have co-written a paper in a specific journal or a conference). In this case, different combinations can expose different information about the authors’ research domain.

We suppose that two authors have published a paper at a data mining conference and another paper at a computer vision conference. Combining the two dimensions may suggest that the research domain of these authors revolves around “image indexing or image clustering”. If the two authors have also co-written an article in a fuzzy logic journal, then we can infer the research domain “fuzzy logic for image processing” by combining the data mining and fuzzy logic dimensions. We can see that different combinations can lead to seemingly divergent predictions. Therefore, a single combination at the initial level can cause a significant loss of information by destroying the divergent information. This information can be useful at a higher level of refinement. For example, we can infer from the initial dimension “fuzzy logic” and the hidden dimension “image indexing or image clustering” that the authors work on “robot navigation”. We illustrate this example in Figure 1a, where nodes 1 and 4 have a high chance of sharing the aforementioned research domains. From this perspective, we can see that some initial dimensions (e.g., “fuzzy logic”) are more informative when they are used to refine higher-level dimensions (e.g., “image clustering”).

**Linear vs Non-Linear:** The high-dimensionality in multiplex graphs gives rise to another phenomenon, which is the complex structure associated with information hidden in a large number of dimensions. We consider a multiplex graph that models a transportation network for a given city. Nodes represent locations in the city, and dimensions represent different means of transportation. An edge in a dimension indicates that two locations are directly linked through a transportation mean. We suppose the existence of three dimensions: Bus Lines ( $G_1$ ), Metro ( $G_2$ ), and Tramway ( $G_3$ ). Fig. 1b shows an illustration of some relevant hidden

dimensions that can be extracted non-linearly from the initial dimensions.

First, we can extract a relevant latent structure  $G'_1$ , which represents the locations that can be reached with two consecutive bus trips. This latent structure can be obtained using a non-linear operation by squaring the adjacency matrix of the Bus Lines dimension  $G_1$ . Since there is a path  $2 \rightarrow 1 \rightarrow 4$  in  $G_1$ , the latent structure  $G'_1$  contains a new link  $2 \rightarrow 4$ . This link is not present in the initial graph structure  $G_1$ .

Another relevant latent structure  $G'_2$  represents the locations that can be accessed by a metro trip followed by a tramway trip. This dimension can be formed by combining the Metro dimension  $G_2$  and the Tramway dimension  $G_3$  using a non-linear operation. More precisely, the links  $1 \rightarrow 4$  in  $G_2$  and  $4 \rightarrow 2$  in  $G_3$  form the link  $1 \rightarrow 2$ . The link  $4 \rightarrow 3$  is formed by  $4 \rightarrow 5$  in  $G_2$  and  $5 \rightarrow 3$  in  $G_3$ .

A third relevant latent structure can be revealed by combining  $G'_1$  and  $G'_2$ . It contains locations that are linked by two bus trips, followed by a metro trip and a tramway trip. This combination forms a new path  $1 \rightarrow 2 \rightarrow 3$ . This path was not present in any of the original dimensions. Most importantly, the link  $2 \rightarrow 3$  cannot be obtained by any linear combination between  $G_1$ ,  $G_2$ , and  $G_3$ .

**Hierarchical Aggregations:** High-dimensional data, such as images or audio, are well-known to be hierarchical in nature [25]. In other words, high-level features are composed of lower-level ones. For example, in image datasets, objects are combinations of motifs, which are themselves combinations of edges. Similarly to high-dimensional data, the complex structure of high-dimensional graphs inherits this compositional aspect. More precisely, there exist high-level hidden dimensions, which can be seen as non-linear compositions of some lower-level dimensions. The two real-world examples presented in Fig. 1a and 1b show the suitability of hierarchical aggregations to account for this compositional aspect.

## 1.3 Contributions

Due to their hierarchical design, neural networks can construct compositions of low-level features and gradually generate high-level patterns [25], [26]. Although the state-of-the-art methods harness the deep learning framework to embed the initial dimensions separately, the aggregation step of these methods remains simple and can not discover the compositional structures hidden in the non-linear combinations of dimensions. To capture complex interactions between the initial dimensions, we advocate the adoption of a hierarchical aggregation method. We take inspiration from the deep learning framework to design our aggregation strategy for high-dimensional multiplex graph embedding, named HMGE (Hierarchical Multiplex Graph Embedding).

Our approach hierarchically combines the dimensions of a multiplex graph to construct new dimensions. Each layer of our model computes a more refined multiplex graph with a lower number of dimensions. The last hidden layer produces a one-dimensional graph. The node embeddings are computed by applying a standard GCN on the graph generated by the last hidden layer. Our training process leverages the hidden dimensions computed at each level

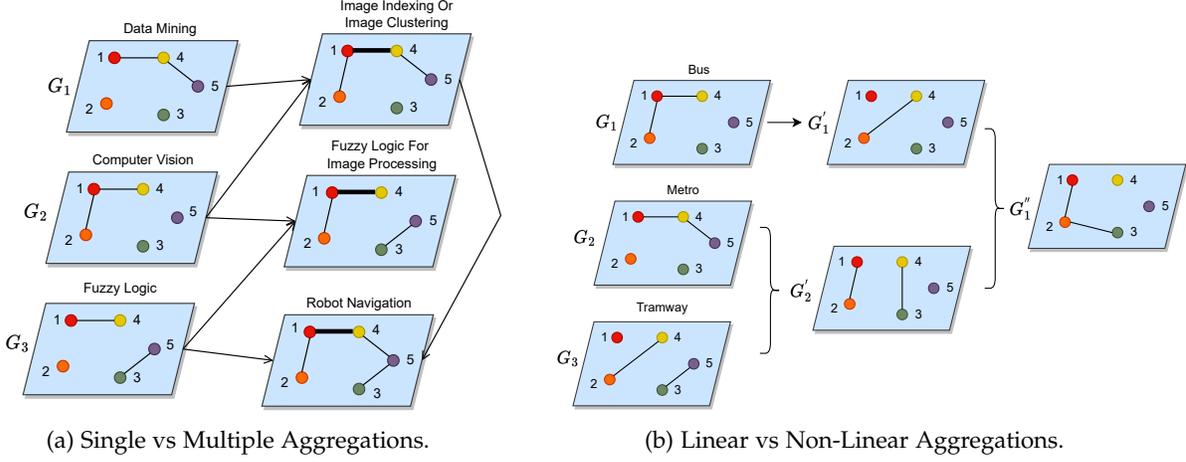


Fig. 1: Motivating examples of hierarchical aggregation of multiplex graphs.

to gradually extract higher-level ones. This progressive refinement allows to alleviate the information loss caused by the widely used single and linear aggregation step. Since non-linear combinations are computed from previous ones, the proposed approach can uncover complex interactions between the different relations. Finally, to encode globally relevant information present in diverse locations of the graph, we introduce mutual information maximization in the optimization module. The significance of this work can be summarized as follows.

- **Methodology:** We propose HMGE, a novel embedding method for high-dimensional multiplex graphs. Our method relies on a hierarchical aggregation strategy to capture complex interactions between the initial dimensions. Progressive refinement of the hidden relations allows to align a large number of divergent and complementary dimensions to a consensus embedding, which in turn alleviates the information loss caused by the widely used single and linear aggregation step.
- **Datasets:** To reflect the compositional nature of high-dimensional multiplex graphs in the experiments, we collected from various sources four multiplex graphs with an important number of dimensions. Specifically, we collected two protein-protein interaction graphs, a co-authorship graph, and a movie database graph. We made this data public for future research in the domain.
- **Experiments:** We conducted detailed experiments on both synthetic and real high-dimensional multiplex graphs. Specifically, we evaluated HMGE on two downstream tasks: node classification and link prediction. Our results show considerable improvement compared to the state-of-the-art methods for several cases. Furthermore, our ablation study confirms that this improvement is imputed to our hierarchical aggregations.

## 2 RELATED WORK

In this section, we review the literature on multiplex graph embedding. Before that, we define two important terms

that we use throughout the section. A *dimension-specific node embedding* is the embedding of a node in a given dimension. A *consensus node embedding* is the embedding of a node taking into account all dimensions [18].

### 2.1 Random Walk-Based Methods

Random walk is a popular approach to graph embedding [27], and several methods have been developed to generalize such a method to the multiplex scenario. SMNE [15] generates random walks in individual dimensions, allowing the method to learn dimension-specific node embeddings. After that, a transformation matrix is applied to obtain consensus node embeddings. MultiVERSE [14] improves SMNE by performing random walks that can jump from one dimension to another. Additionally, it supports the presence of heterogeneous nodes in the multiplex graph. PMNE [16] designs first- and second-order random walks to browse single layers. It uses the Node2Vec [28] parameters  $p$  and  $q$  to flexibly traverse the neighbors of the node.

A drawback of random walk-based methods is that they are inherently transductive (that is, they cannot handle the arrival of unobserved nodes). GATNE [29] solves this issue by generating training samples using random walks and then computing node embeddings with trainable transformations. In the case of high-dimensional multiplex graphs, random walk-based methods suffer from a major limitation. Indeed, to cover all the graph dimensions, long sequences of random walks must be generated. Since the skip-gram model takes advantage of local patches to optimize representations [30], it can be difficult to capture long-range dependencies in the node sequences. In other words, these methods cannot effectively uncover latent structures spanning multiple dimensions. Moreover, random walk-based methods lack the expressive power to model the compositional nature of high-dimensional multiplex graphs.

### 2.2 Sharing Information with Regularization

While learning node embeddings, sharing information between dimensions is essential to capture common and complementary information. To this end, a handful of methods rely on regularization. DMNE [19] uses an auto-encoder to

encode each dimension of the multiplex graph. After that, a regularization term is applied to force the dimension-specific embeddings to be similar. This results in a consensus embedding for each node. MELL [20] follows a similar approach, but for directed multiplex graphs. It uses two embedding vectors for each node: one as a head vector and another as a tail vector. In addition, MELL learns layer-specific representations (i.e., embeddings for entire layers). These representations are used to compute the probability of an edge between two nodes in a specific layer. The optimized loss function is computed from these probabilities, in a similar fashion to the first-order loss of LINE [31]. MTNE [18] also uses a regularization term to encode the different dimensions to the consensus embeddings. In contrast to the other methods, the dimension-specific embeddings are generated from a pretraining stage.

Sharing information with regularization is a simple technique to build consensus embeddings. In fact, most of the regularization-based methods force the dimension-specific embeddings to be similar. However, the used similarity metrics are fixed and simple. Without prior knowledge to systematically identify the best metric, these methods generally underfit the complex interactions between the different dimensions. More precisely, these approaches cannot account for the compositional structures hidden in hierarchical combinations of the initial dimensions. Consequently, the consensus embeddings obtained based on these methods can only contain *low-level* and *common* information. In the presence of complementary dimensions, forcing the embeddings to be similar can cause significant information losses, which in turn hinders the quality of the final representations. Because of these limitations, our approach does not rely on regularization. Instead, it learns to hierarchically combine the multiplex graph dimensions to extract both *common* and *complementary high-level* information.

### 2.3 Multiplex Graph Neural Networks

Learning dimension-specific node representations using GNNs and then combining these representations based on simple aggregations (e.g., linear combination, concatenation) is the gold standard for the most recent body of literature. For instance, DMGI [11] computes consensus embeddings by assigning attention weights to the node representations of each dimension. The training process of DMGI leverages InfoMax (mutual information maximization) [32] to optimize the latent representations. HDMI [12] improves on DMGI by defining a higher-order InfoMax mechanism that includes node features. SSSDCM [33] applies InfoMax between local node-level representations and global cluster-aware graph summary.

A limitation of these methods is the loss of information that results from the single and linear aggregation of dimension-specific node embeddings. Different combinations of dimensions can highlight different interactions between dimensions [24]. This means that some information can only be uncovered by combining specific sets of dimensions. Second, these methods do not account for the compositional nature of high-dimensional multiplex graphs. Consequently, they cannot learn hidden structures that capture complex interactions between different dimensions. On the

contrary, our approach hierarchically combines the initial dimensions, making it possible to retrieve latent structures at different levels of the hierarchy. Moreover, our method learns more complex combinations than linear aggregation methods, which we will show in Section 3.4.

## 3 PROPOSED APPROACH

Let us first present the notations used throughout the paper and define the problem of multiplex graph embedding.

**Multiplex Graph:** A multiplex graph is defined as a set of  $D$  graphs  $G = (G_1, G_2, \dots, G_D)$ , where  $G_d = (V, A_d, X)$  is a graph with  $N$  nodes from the set  $V = \{v_1, v_2, \dots, v_N\}$ ,  $A_d \in \{0, 1\}^{N \times N}$  is the adjacency matrix and  $X \in \mathbb{R}^{N \times F}$  is the node feature matrix. The nodes and their features are shared across all graphs, while each  $G_d$  has its own adjacency matrix. We refer to each graph  $G_d$  as a dimension of  $G$ .

**Multiplex Graph Embedding:** Given  $G$ , the goal is to learn an  $M$ -dimensional vector representation  $z_i \in \mathbb{R}^M$  for each node  $v_i \in V$ , in an unsupervised setting. The vector representations are regrouped in a matrix  $Z \in \mathbb{R}^{N \times M}$ , which can be used in multiple downstream tasks, such as node classification and link prediction.

HMGE learns node embeddings by computing trainable non-linear combinations of the initial dimensions. More precisely, the graph dimensions are hierarchically combined; that is, each new combination is computed based on combinations from a lower level. Each dimension  $G_d$  is represented by its corresponding adjacency matrix  $A_d$ . New combinations are represented by new adjacency matrices resulting from hierarchical aggregations. This process forms a hierarchy of latent dimensions that reflect the compositional aspect of high-dimensional graphs. In fact, hierarchical aggregation aims to encode complex interactions between the initial dimensions in the final embeddings. To train the model, we maximize the mutual information between the graph-level representation and the local patches.

In this section, we give a detailed presentation of HMGE and of the hierarchical aggregation process. After that, we show formally that our hierarchical aggregation strategy can capture more complex patterns than linear aggregation methods such as DMGI [11] and HDMI [12].

### 3.1 Overview of HMGE

Fig. 2 depicts the architecture of the proposed approach. HMGE introduces a graph neural network with  $L$  layers that can tackle the multiplex case. Unlike previous methods, our approach can learn both latent features and latent graph structures. Specifically, each layer  $l$  takes as input a multiplex graph  $G^{(l-1)}$ , whose attributes are the node embeddings  $H^{(l-1)}$  ( $G^{(0)} = G$  and  $H^{(0)} = X$ ). Then, this layer outputs a new multiplex graph  $G^{(l)}$ , whose attributes  $H^{(l)}$  and graph structures  $A_d^{(l)}$  are computed from  $G^{(l-1)}$ .

Fig. 2a illustrates HMGE on a three-dimensional multiplex graph. The first layer computes the node embedding matrix  $H^{(1)}$  and transforms the three input dimensions into two dimensions by combining the graph structures of  $G_1$  and  $G_2$ . The second layer computes  $H^{(2)}$  by refining  $H^{(1)}$  using the newly created dimensions and combines the

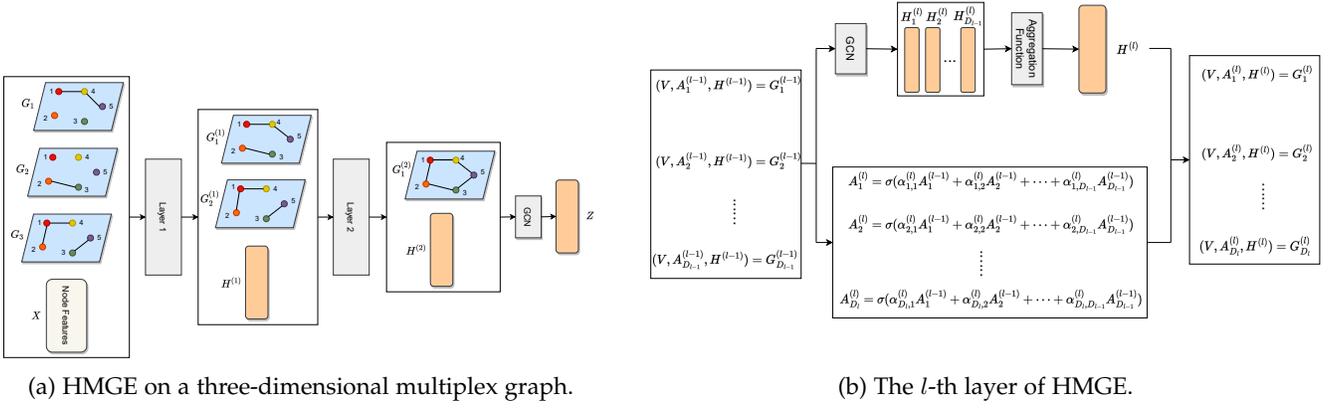


Fig. 2: The architecture of HMGE.

structures of  $G_1^{(1)}$  and  $G_2^{(1)}$  to obtain a single graph  $G_1^{(2)}$ . The final node embeddings  $Z$  are computed by running a graph convolutional layer (GCN) on  $G_1^{(2)}$ .

Multiple transformations are applied inside each layer  $l$ , as depicted in Fig. 2b. More precisely, each layer operates in two phases: 1) computing the node embedding matrix  $H^{(l)}$ ; 2) generating hidden dimensions in the form of a multiplex graph  $G^{(l)}$ . In the next subsections, we present in detail how the embeddings are computed and how the hierarchical aggregation strategy refines the initial multiplex graph structures to obtain lower-dimensional multiplex graph structures.

### 3.2 Computing Node Embeddings

The first phase of the encoding process in each layer  $l$  focuses on computing the node embeddings. Given the representation  $H^{(l-1)}$  and the graph structures  $A_d^{(l-1)}$ , a single graph convolutional operation is applied on each dimension  $d$  of  $G^{(l-1)}$  to obtain new node representations  $H_d^{(l)}$ :

$$H_d^{(l)} = \text{ReLU}(\hat{\Delta}_d^{-\frac{1}{2}} \hat{A}_d^{(l-1)} \hat{\Delta}_d^{-\frac{1}{2}} H^{(l-1)} W_d^{(l)}), \quad (1)$$

where  $\hat{A}_d^{(l-1)} = A_d^{(l-1)} + I$ ,  $\hat{\Delta}_d = \text{diag}(\hat{A}_d^{(l-1)} \mathbf{1}_N)$ ,  $\mathbf{1}_N \in \mathbb{R}^N$  is a vector of ones,  $W_d^{(l)}$  is the weight matrix of the graph convolutional layer, and  $\text{ReLU}$  is the rectified linear unit activation function. After that, the dimension-specific embeddings  $H_d^{(l)}$  are aggregated into a single embedding  $H^{(l)}$  as described by the following equations:

$$h_n^{(l)} = \sum_{d=1}^{D_{l-1}} \beta_{n,d}^{(l)} h_{n,d}^{(l)}, \quad (2)$$

$$H^{(l)} = \left\|_{n=1}^N h_n^{(l)}, \quad (3)$$

where  $h_n^{(l)}$  is the embedding of node  $v_n$  generated by the  $l^{\text{th}}$  layer,  $\beta_{n,d}^{(l)}$  are attention weights [34],  $D_{l-1}$  denotes the number of dimensions in  $G^{(l-1)}$ , and  $\left\|$  stands for the concatenation operation. The matrix  $H^{(l)}$  constitutes the node features of the newly generated multiplex graph  $G^{(l)}$ , which is the input of the next layer of HMGE.

The importance of each dimension in  $G^{(l-1)}$  is measured by the attention weights  $\beta_{n,d}^{(l)}$  which are computed as follows:

$$\hat{\beta}_{n,d}^{(l)} = \tanh(y_d^{(l)T} V_d^{(l)} h_{n,d}^{(l)}), \quad (4)$$

$$\beta_{n,d}^{(l)} = \frac{\hat{\beta}_{n,d}^{(l)}}{\sum_{d'=1}^{D_{l-1}} \hat{\beta}_{n,d'}^{(l)}}, \quad (5)$$

where  $V_d^{(l)} \in \mathbb{R}^{M \times M}$  and  $y_d^{(l)} \in \mathbb{R}^M$  are part of the training parameters. The attention weights adjust the contribution of each individual dimension to the representations. In spite of its widespread adoption, the attention mechanism is not sufficient to capture complex interactions between the graph dimensions. To address this limitation, the second phase of the encoding process at each layer refines the graph structures.

### 3.3 Hierarchical Aggregations

The second phase of the encoding process in each layer  $l$  focuses on computing the graph structures  $A_d^{(l)}$ . As discussed in the motivation subsection, relevant information can be hidden in non-linear compositions of the graph dimensions. To identify this information, the  $l^{\text{th}}$  layer computes  $D_l$  output adjacency matrices from  $D_{l-1}$  input adjacency matrices such that  $D_{l-1} < D_l$ .

Each output graph structure is computed based on a weighted summation of the input adjacency matrices followed by a non-linear activation function. The weighted summation can uncover hidden paths in the multiplex graph. It also transforms multiple adjacency matrices into a single output matrix. The activation function models non-linear interactions, and stacking multiple HMGE layers can capture more complex interactions. Accordingly, the  $j^{\text{th}}$  output adjacency matrix of the  $l^{\text{th}}$  layer is computed as follows:

$$A_j^{(l)} = \sigma\left(\sum_{i=1}^{D_{l-1}} \alpha_{i,j}^{(l)} A_i^{(l-1)}\right), \quad (6)$$

where  $\sigma$  is an activation function (we use  $\text{ReLU}$  in the experiments), and  $\alpha_{i,j}^{(l)}$  is the weight associated with the

$i^{\text{th}}$  input and  $j^{\text{th}}$  output dimensions. Let  $\alpha^{(l)} = (\alpha_{i,j}^{(l)}) \in \mathbb{R}^{D_{l-1} \times D_l}$  be the matrix that contains the combination weights. Since  $\alpha^{(l)}$  is a trainable matrix, HMGE learns combinations of adjacency matrices that maximize the mutual information objective function. Note that before computing Eq. (6), the weights  $\alpha_{i,j}^{(l)}$  are normalized with a softmax function:

$$\alpha_{i,j}^{(l)} = \frac{\exp(\alpha_{i,j}^{(l)})}{\sum_{k=1}^{D_{l-1}} \exp(\alpha_{k,j}^{(l)})}. \quad (7)$$

The  $l^{\text{th}}$  layer learns and computes multiple combinations, which are then used by the layer  $l + 1$  to improve the node embeddings  $H^{(l)}$  and compute other combinations. This process forms a hierarchy of adjacency matrices as illustrated in Fig. 2a. In addition, since each layer increments on the embeddings of the previous layer, the final representation  $Z$  contains information about both the input graph  $G$  and the intermediate graphs  $G^{(l)}$ . Moreover, by applying an activation function on the output dimensions and stacking multiple layers, HMGE can uncover non-linear latent structures hidden in the input multiplex graph dimensions.

To tackle the high-dimensionality of real-world multiplex graphs, the proposed solution gradually reduces the number of dimensions of the input at each layer. When the number of dimensions is high, aggregating all embeddings in a single step is ineffective, as it can cause a significant loss of information. HMGE addresses this issue by aggregating the embeddings multiple times on several intermediate multiplex graphs. We show in the next subsection that this hierarchical aggregation process can model more complex patterns than linear aggregation.

### 3.4 HMGE vs Linear Aggregation Methods

Most current approaches to multiplex graph embedding rely on linear aggregation to extract node representations [11], [12]. More specifically, they compute node embeddings on individual dimensions and then aggregate them using a weighted summation. Attention weights are used to account for the relevance of each dimension. In this section, we show that hierarchical aggregations can model more complex patterns than linear aggregation approaches.

For readability, we ignore the attention weights in the following equations and suppose that all graph convolutional operations have the same weight matrix  $W$ . Moreover, we ignore the activation functions to clearly exhibit the combinations that are formed by the different methods. For illustration purposes, consider a two-dimensional multiplex graph in the input. The following equation summarizes the embedding module of a linear aggregation method with two layers:

$$\begin{aligned} Z &= A_1 (A_1 X W) W + A_2 (A_2 X W) W, \\ &= (A_1^2 + A_2^2) X W^2. \end{aligned} \quad (8)$$

We can see that  $A_1^2$  and  $A_2^2$  are summed to compute the node embedding matrix  $Z$ . On the other side, a two-layer HMGE embedding module on the same input graph can be summarized as follows:

$$\begin{aligned} Z &= (\alpha_1 A_1 + \alpha_2 A_2) H^{(1)} W, \\ &= (\alpha_1 A_1 + \alpha_2 A_2) (A_1 X W + A_2 X W) W, \\ &= (\alpha_1 A_1 + \alpha_2 A_2) (A_1 + A_2) X W^2, \\ &= (\alpha_1 A_1^2 + \alpha_1 A_1 A_2 + \alpha_2 A_2 A_1 + \alpha_2 A_2^2) X W^2, \\ &= (\alpha_1 A_1^2 + (\alpha_1 + \alpha_2) A_1 A_2 + \alpha_2 A_2^2) X W^2. \end{aligned} \quad (9)$$

HMGE can learn more complex patterns than linear aggregation methods. Using hierarchical aggregations, both  $A_1^2$  and  $A_2^2$  are leveraged, while also introducing  $A_1 A_2$ . Moreover, the weights  $\alpha_1$  and  $\alpha_2$  make it possible to consider the importance of each term. The term  $A_1 A_2$  may reveal edges that cannot be retrieved by considering  $A_1$  and  $A_2$  individually, or by combining them linearly. For example, as illustrated in Fig. 1b, the combination of two adjacency matrices (specifically, in this figure, the product of the matrices) forms relevant paths between nodes that were not previously connected.

In practice, given a high-dimensional multiplex graph, several HMGE layers can be stacked to construct even more complex combinations. In the experiments section, we show that the additional expressive power of HMGE improves the performance on downstream tasks compared to linear aggregation methods on both synthetic and real-world data.

### 3.5 Training Algorithm

HMGE is trained based on a self-supervised strategy [35], which involves the learning of general-purpose node embeddings. Therefore, the trained model can be used to perform multiple downstream tasks. Specifically, inspired by Deep Graph Infomax (DGI) [36], we employ a loss based on mutual information maximization [32]. The training procedure maximizes the mutual information between the graph-level representation and the local patches from the set  $\{z_1, z_2, \dots, z_N\}$ . The graph-level representation  $s$  is a summary of the entire graph  $G$  and is obtained by aggregating all the node embeddings  $z_i$ :

$$s = \frac{1}{N} \sum_{i=1}^N z_i. \quad (10)$$

A discriminator  $\mathcal{D}$  is used as a proxy to maximize the intractable mutual information function. The discriminator takes as input a patch-summary pair and is trained to compute the probability score assigned to the pair. To this end, we select positive pairs, which consist of the graph summary  $s$  and samples from  $Z = HDME(G)$ . On the other hand, negative pairs are formed from  $s$  and samples from  $\hat{Z} = HDME(\hat{G})$ . The graph  $\hat{G}$  is a corrupted version of the input graph that can be generated in multiple ways, such as swapping nodes and removing or adding links. In this work, we randomly shuffle the node features to obtain a corrupted version  $\hat{X}$  and use this version as the feature matrix of  $\hat{G}$ . The discriminator function  $\mathcal{D}$  is implemented based on bilinear scoring:

$$\mathcal{D}(h_i, s) = \text{Sigmoid}(h_i Q s), \quad (11)$$

---

**Algorithm 1** *HMGE*

---

**Input:** multiplex graph  $G$ , embedding dimension  $M$ , number of layers  $L$ , number of iterations  $T$ .  
**Output:** Node embeddings  $Z$ .

- 1: **for**  $epoch \leftarrow 1$  to  $T$  **do**
- 2:  $\hat{X} \leftarrow \text{Shuffle}(X)$
- 3:  $\hat{G} \leftarrow G$ , but replace  $X$  with  $\hat{X}$ .
- 4:  $Z \leftarrow \text{Encode-Multiplex-Graph}(G)$
- 5:  $\hat{Z} \leftarrow \text{Encode-Multiplex-Graph}(\hat{G})$
- 6:  $s \leftarrow \frac{1}{N} \sum_{i=1}^N z_i$
- 7:  $\mathcal{L} \leftarrow \sum_{z_i \in Z} \log \mathcal{D}(z_i, s) + \sum_{\hat{z}_j \in \hat{Z}} \log(1 - \mathcal{D}(\hat{z}_j, s))$
- 8: Update  $\{\alpha^{(l)}, W_d^{(l)}, V_d^{(l)}, y_d^{(l)}\}$  to minimize  $\mathcal{L}$  with Adam.
- 9: **end for**
- 10: **return**  $Z$ .

---

**Algorithm 2** *Encode-Multiplex-Graph*

---

**Input:** multiplex graph  $G$ , number of layers  $L$ .  
**Output:** node embeddings  $H^{(L)}$ .

- 1:  $G^{(0)} \leftarrow G$
- 2:  $H^{(0)} \leftarrow X$
- 3: **for**  $l \leftarrow 1$  to  $L$  **do**
- 4:  $G^{(l)}, H^{(l)} \leftarrow \text{Hierarchical-Aggregation}(G^{(l-1)}, H^{(l-1)})$
- 5: **end for**
- 6: **return**  $H^{(L)}$ .

---

where  $Q \in \mathbb{R}^{M \times M}$  is a parameter matrix, and  $M$  is the size of node embeddings. We minimize the binary cross-entropy loss  $\mathcal{L}$  to train the discriminator and generate the node embeddings as described by:

$$\mathcal{L} = \sum_{i=1}^N \log \mathcal{D}(z_i, s) + \sum_{j=1}^N \log(1 - \mathcal{D}(\hat{z}_j, s)). \quad (12)$$

Algorithms 1, 2 and 3 summarize the proposed approach. We train the model for  $T$  iterations and update the parameters  $\{\alpha^{(l)}, W_d^{(l)}, V_d^{(l)}, y_d^{(l)}\}$  by minimizing the loss  $\mathcal{L}$  using the Adam optimizer. The time complexity of HMGE is  $\mathcal{O}(TLD(\mathcal{E}M + NM^2 + \mathcal{E}D))$ , where  $N$  is the number of nodes,  $D$  the number of input dimensions,  $L$  the number of embedding layers,  $M$  the size of node embeddings, and  $\mathcal{E}$  is the maximum number of edges in the graph dimensions. The memory complexity of our approach is  $\mathcal{O}(LD(\mathcal{E} + NM + M^2 + 2D^2))$ .

## 4 EXPERIMENTS

In this section, we evaluate the suitability of HMGE for high-dimensional multiplex graph embedding. First, we perform experiments on synthetically generated data. After that, we compare HMGE with various multiplex graph embedding methods on real-world data for two downstream tasks: link prediction and node classification. Finally, we present an ablation study followed by visualizations of the embeddings and combination weights  $\alpha^{(l)}$ .

---

**Algorithm 3** *Hierarchical-Aggregation*

---

**Input:** multiplex graph  $G^{(l-1)}$ , node embeddings  $H^{(l-1)}$ .  
**Output:** multiplex graph  $G^{(l)}$ , node embeddings  $H^{(l)}$ .

- 1:  $H^{(l)} \leftarrow \text{Phase-One}(G^{(l-1)}, H^{(l-1)})$
- 2:  $G^{(l)} \leftarrow \text{Phase-Two}(G^{(l-1)})$
- 3: **return**  $G^{(l)}, H^{(l)}$ .

- 4: **Function**  $\text{Phase-One}(G^{(l-1)}, H^{(l-1)})$
- 5: **for**  $d \leftarrow 1$  to  $D_{l-1}$  **do**
- 6:  $H_d^{(l)} \leftarrow \text{GCN}(H^{(l-1)}, A_d^{(l-1)})$
- 7: **end for**
- 8: Compute  $\beta^{(l)}$  according to Eq. (4) and Eq. (5).
- 9:  $h_n^{(l)} = \sum_{d=1}^{D_{l-1}} \beta_{n,d}^{(l)} h_{n,d}^{(l)}$
- 10:  $H^{(l)} = \left\| \right\|_{n=1}^N h_n^{(l)}$
- 11: **return**  $H^{(l)}$ .

- 12: **Function**  $\text{Phase-Two}(G^{(l-1)})$
- 13: **for**  $j \leftarrow 1$  to  $D_l$  **do**
- 14:  $A_j^{(l)} \leftarrow \sigma(\sum_i^{D_{l-1}} \alpha_{i,j}^{(l)} A_i^{(l-1)})$
- 15:  $G_j^{(l)} \leftarrow (V, A_j^{(l)}, H^{(l)})$
- 16: **end for**
- 17:  $G^{(l)} \leftarrow (G_1^{(l)}, G_2^{(l)}, \dots, G_{D_l}^{(l)})$
- 18: **return**  $G^{(l)}$ .

---

## 4.1 Experimental Setup

### 4.1.1 Datasets

Due to the scarcity of real-world labeled high-dimensional multiplex graphs, we collected from various sources four datasets to assess the suitability of HMGE. Table 1 summarizes the statistics of these multiplex graphs. The datasets and the implementation of HMGE are on Github <sup>2</sup>.

**BIOGRID:** We collected a multiplex graph from the BIOGRID database of proteins [2]. Nodes represent proteins, and edges represent protein-protein interactions. Protein-protein interactions can be inferred by various protocols, for example, by using the biochemical effect of one protein on another or from X-ray Crystallography at the atomic level [2]. The dimensions of the multiplex constitute different inference protocols. Node classification labels represent the species from which the proteins have been extracted.

**DBLP-Authors:** This dataset represents a coauthorship graph. We used AMiner [37] to collect it. The nodes are the authors of research papers, and an edge between two authors indicates that they have co-written a paper. Dimensions represent various conferences and journals. For example, if two authors are connected in dimension  $d_1$ , they have co-written a paper in the conference or journal  $d_1$ . We use the authors' research areas as classification labels. Note that we end up with a multilabel classification problem (i.e., a node can belong to more than one class).

**IMDB:** It is a multiplex graph extracted from IMDB <sup>1</sup>. Nodes are movies, and an edge between two movies indicates that a person has participated in both movies. Different dimensions represent different roles: actors, direc-

2. <https://github.com/abdouskamel/HMGE>  
1. <https://www.imdb.com/interfaces/>

Dataset	# Dimensions	# Nodes	# Edges	# Training data	# Classes
BIOGRID	15	4,211	280 979	252	4
DBLP-Authors	10	5,124	33 250	307	4
IMDB	8	3000	224 984	173	3
STRING-DB	7	4,083	4 923 554	244	3

TABLE 1: Real-world data statistics.

tors, producers, etc. We use the movie genre as classification labels.

**STRING-DB:** This is another protein-protein interaction graph collected from STRING-DB [38]. It is similar to BIOGRID in terms of node, edge, and dimension semantics. The difference with BIOGRID is in the classification labels. In this dataset, the labels represent protein families instead of animal species.

#### 4.1.2 Compared Algorithms

Our comparison focuses on methods designed for multiplex graph embedding.

**CrossMNA [13]:** This method defines node embeddings as a combination of intra-vectors (i.e., dimension-specific embeddings) and inter-vectors (i.e., consensus embeddings). It then optimizes a loss based on edge reconstruction.

**MultiVERSE [14]:** It is a random walk-based method. It learns embedding by performing random walks on the supra-adjacency matrix of the multiplex graph (i.e., walks can jump from one dimension to another). After that, the random walks are turned into a similarity matrix from which node embeddings are then generated.

**GATNE [29]:** This method runs random walks on each dimension of the graph and then generates new training samples based on nodes’ proximity. It then computes embeddings with a trainable linear combination mixed with attention. Finally, it optimizes a loss function based on graph reconstruction.

**DMGI [11]:** It is a linear aggregation method that learns individual node embeddings in each dimension by maximizing mutual information. After that, embeddings are aggregated using attention scores. A regularization term is employed to fine-tune the representations.

**HDMI [12]:** It is an extension of DMGI that defines a high-order mutual information loss. On each dimension, it maximizes the mutual information between a graph summary, local patches, and node features.

**SSDCM [33]:** Similar to DMGI and HDMI, this method is based on mutual information maximization. However, structure-aware global graph representations are leveraged to optimize the InfoMax loss. This is achieved by a cluster-aware strategy for graph summary generation.

#### 4.1.3 Evaluation Protocol

To evaluate HMGE, we perform both link prediction and node classification. We use the unsupervised loss in Eq. (12) to learn the node embeddings for both tasks. Then, for node classification, we run logistic regression on the embeddings and evaluate the results with F1-Macro and F1-Micro. For link prediction, we use the inner product activated with a Sigmoid function to calculate the missing links scores:  $\text{Sigmoid}(ZZ^T)$ . The results are then evaluated with the area under the ROC curve (AUC-ROC) and average precision

(AP). Note that for link prediction, we randomly remove 10% of the links for the sake of evaluation.

#### 4.1.4 Parameter Settings

For our method, we set the size of node embeddings to 64 (except on BIOGRID, where we set it to 32, as it leads to better results) and the number of layers to 2. We use Adam as an optimizer with a learning rate of 0.001 and a weight decay of  $10^{-5}$ . We train for 2,000 epochs and adopt early stopping with a patience of 100.

## 4.2 Experiments on Synthetic Data

We first compare HMGE with DMGI and HDMI on generated data. Through this evaluation, we show that when the graph dimensions increase, the classification accuracy of the competitors rapidly decreases. In contrast, the accuracy of HMGE is significantly less affected by the increase in the number of dimensions.

#### 4.2.1 Data Generation

To synthesize a multiplex graph with  $N$  nodes,  $D$  dimensions and  $K$  classes, we run a stochastic block model (SBM) [39]  $D$  times. Each run generates a dimension of the graph along with node labels specific to this dimension. The SBM proceeds as follows: (1) Each node has a probability  $p_i$  to belong to class  $c_i$ . We set  $K = 2$  (binary classification) with  $p_1 = p_2$ ; (2) For every pair of nodes  $v_i$  and  $v_j$ , such that  $v_i$  is in class  $c_i$  and  $v_j$  is in class  $c_j$ , the edge  $(v_i, v_j)$  has a probability  $p_{i,j}$ . We use two probabilities  $p_{in}$  and  $p_{out}$ : the probabilities of an edge between two nodes of the same class and of different classes, respectively.

After running SBM  $D$  times, we obtain  $D$  adjacency matrices each with its own classification labels. We use a voting mechanism across the dimensions to get a single global label for each node. For example, if  $v$  is in class  $c_1$  in dimension  $G_1$ , in class  $c_2$  in dimension  $G_2$ , and in class  $c_1$  in dimension  $G_3$ , we assign  $v$  to class  $c_1$ . This process creates a multiplex graph whose dimensions contribute to the node labels. In this context, an effective embedding method should capture as much information as possible from every dimension.

In total, we generate 9 synthetic multiplex graphs with 3,000 nodes and with the following number of dimensions:  $\{3, 7, 11, 15, 21, 25, 31, 35, 41\}$ . We set  $p_1 = p_2 = 0.5$ ,  $p_{in} = 0.05$ , and  $p_{out} = 0.01$ . We generate binary classes and use the classification accuracy for evaluation.

#### 4.2.2 Evaluation Results

Fig. 3 shows the node classification results on the synthetic datasets. We can see that the accuracy of DMGI and HDMI significantly decreases as the number of dimensions increases, going from a near-perfect classification when  $D = 3$

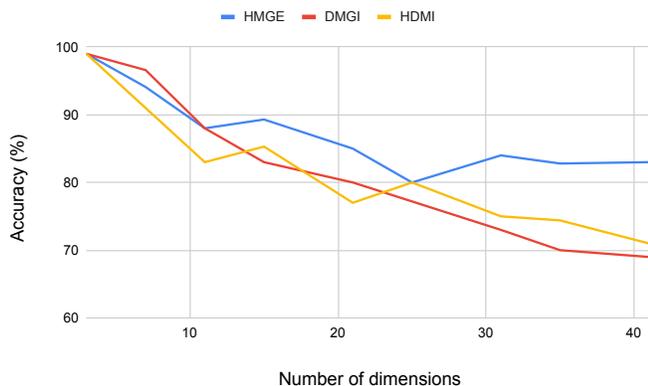


Fig. 3: Node classification results on synthetic data.

to below 70% when  $D = 41$ . As the number of dimensions increases, the global node labels become more and more different from the dimension-specific labels, making predicting the correct global node labels more difficult. On the other hand, the accuracy of HMGE drops at a considerably slower rate, reaching 83% when  $D = 41$ .

Note that the competing algorithms have near-perfect accuracy when  $D = 3$ . This is because each dimension has a simple structure consisting of two highly dense regions connected by a sparse region. When  $D$  is small, predicting the node labels from this simple structure is straightforward, but the prediction becomes more difficult as  $D$  increases. These results illustrate the difficulty of effectively encoding high-dimensional multiplex graphs.

Furthermore, the results suggest that competitors considered in the comparison do not fully exploit the information provided by each dimension. This aspect exhibits a serious limitation of state-of-the-art methods, particularly common embedding techniques that use linear aggregation. HMGE alleviates this issue by leveraging hierarchical aggregations.

### 4.3 Experiments on Real-World Data

In this section, we evaluate HMGE on real-world high-dimensional multiplex graphs collected from various sources (Table 1). We perform two downstream tasks: link prediction and node classification.

#### 4.3.1 Link Prediction

Table 2 shows a comparison between the proposed approach and state-of-the-art methods on the link prediction task. We can see that most compared algorithms have AUC and AP scores below or near 50%. These methods are not well-suited for link prediction on high-dimensional multiplex graphs for several reasons. On the one hand, MultiVERSE and GATNE require long sequences of random walks to consider all dimensions. Moreover, these methods can not capture long-range dependencies in the node sequences. On the other hand, SSDCM, DMGI, and HDMI are limited by the single and linear aggregation of dimension-specific node embeddings. Therefore, they fail to capture the compositional relations between the initial dimensions.

We can also see from Table 2 that HMGE yields better results than the compared algorithms. In particular, our method outperforms the state-of-the-art models by a significant margin in several cases. For instance, the difference in the link prediction task between HMGE and the most competitive method on BIOGRID (i.e., CrossMNA) is higher than 20% in terms of AUC. Furthermore, HMGE achieves AUC and AP scores over 70% on BIOGRID, and close to 70% on DBLP-Authors. These two datasets have the highest number of dimensions among the other ones. Our results substantiate the effectiveness of HMGE in encoding high-dimensional multiplex graphs. Unlike previous methods, our approach stands out by its ability to perform hierarchical aggregations, which are introduced to capture compositional interactions between the initial dimensions.

#### 4.3.2 Node Classification

Table 3 shows the evaluation results on node classification. HMGE outperforms HDMI, DMGI, and SSDCM, which are based on mutual information maximization and linear aggregation. Particularly, these methods are less competitive on BIOGRID, which is the dataset with the highest number of dimensions. For instance, the difference in node classification performance between HMGE and DMGI on BIOGRID is higher than 40% in terms of F1-Macro and F1-Micro. As the number of dimensions increases, the linear aggregation strategy becomes less effective. We reported similar results on synthetic data. In the high-dimensional setting, informative and complex latent structures can be hidden across various dimensions. These latent structures can be established hierarchically from the dimension-specific embeddings.

We can also see from Table 3 that our model yields better results than random walk-based methods (GATNE and MultiVERSE). These methods use the skip-gram model, which has limited capacity to capture long-range dependencies in the node sequences. Unlike these methods, HMGE leverages hierarchical aggregations to capture the compositional interactions between the initial dimensions in the final node embeddings.

### 4.4 Ablation Study

In this section, we perform two ablation experiments to show the significance of our contributions: ablation of the whole hierarchical aggregation mechanism, and ablation of the combination weights.

#### 4.4.1 First Ablation (Hierarchical Aggregation Mechanism)

We evaluate the performance of HMGE without using the hidden layers that generate the latent multiplex graphs. When the number of hidden layers is zero, our training process amounts to learning the embeddings on each dimension with GCNs and then aggregating them linearly to a single representation. The goal is to illustrate the benefits of hierarchical aggregations to high-dimensional multiplex graph embedding.

The first rows of Tables 4 and 5 show the results obtained after performing the first ablation, respectively on the tasks of link prediction and node classification. We can see that the hierarchical aggregation mechanism significantly improves

Dataset	BIOGRID		DBLP-Authors		IMDB		STRING-DB	
Metrics	AUC	AP	AUC	AP	AUC	AP	AUC	AP
CrossMNA	50.95	50.48	53.17	51.63	50.9	50.45	50.19	50.09
MultiVERSE	50.3	50.15	54.54	52.38	48.34	49.15	50	50
GATNE	39.82	42.53	43.57	44.34	37.54	41.05	48.15	47.89
SSDCM	32.14	40.52	<u>62.74</u>	58.54	<u>53.84</u>	<u>54.48</u>	50.4	50.2
DMGI	42.33	44.16	50	50	52.84	52.66	<u>54.62</u>	52.69
HDMI	43.91	46.01	60.64	<u>59.36</u>	50.95	50.76	54.46	<u>53.96</u>
HMGE	<b>71.76</b>	<b>70.17</b>	<b>67.31</b>	<b>67.21</b>	<b>57.42</b>	<b>55.77</b>	<b>65.46</b>	<b>62.84</b>

TABLE 2: Results on link prediction. Best in bold and second best underlined.

Dataset	BIOGRID		DBLP-Authors		IMDB		STRING-DB	
Metrics	F1-Macro	F1-Micro	F1-Macro	F1-Micro	F1-Macro	F1-Micro	F1-Macro	F1-Micro
CrossMNA	97.04	96.98	<b>63.52</b>	70.08	34.42	34.33	33.26	72.44
MultiVERSE	<u>98.53</u>	<u>98.53</u>	57.9	60.72	41.3	41.3	46.68	47.75
GATNE	98.43	98.49	<u>58.12</u>	71.34	<u>42.52</u>	<u>42.31</u>	70.1	72.11
SSDCM	11.22	28.42	<u>57.67</u>	<u>71.70</u>	24.78	33.59	61.13	65.84
DMGI	50.98	51.96	54.49	62.36	38.2	38.2	65.61	67.62
HDMI	62.9	64.46	57.1	70.8	38.9	39.5	<u>72.03</u>	<u>73.94</u>
HMGE	<b>98.75</b>	<b>98.77</b>	57.52	<b>71.76</b>	<b>43.02</b>	<b>43.16</b>	<b>80.33</b>	<b>82.08</b>

TABLE 3: Results on node classification. Best in bold and second best underlined.

Dataset	BIOGRID		DBLP-Authors		IMDB		STRING-DB	
Metrics	AUC	AP	AUC	AP	AUC	AP	AUC	AP
HMGE (First Ablation)	47.16	47.17	<u>56.02</u>	<u>53.76</u>	41.73	43.70	59.75	58.23
HMGE (Second Ablation)	<u>66.48</u>	<u>63.38</u>	47.91	49.05	52.45	51.28	<u>62.64</u>	<u>60.02</u>
HMGE	<b>71.76</b>	<b>70.17</b>	<b>67.31</b>	<b>67.21</b>	<b>57.42</b>	<b>55.77</b>	<b>65.46</b>	<b>62.84</b>

TABLE 4: Ablation study on the task of link prediction. Best in bold and second best underlined.

Dataset	BIOGRID		DBLP-Authors		IMDB		STRING-DB	
Metrics	F1-Macro	F1-Micro	F1-Macro	F1-Micro	F1-Macro	F1-Micro	F1-Macro	F1-Micro
HMGE (First Ablation)	91.07	91.26	56.74	70.07	<u>39.18</u>	<u>40.97</u>	61.95	66.69
HMGE (Second Ablation)	<u>95.87</u>	<u>96.03</u>	<u>57.35</u>	<u>70.35</u>	21.29	34.75	<u>79.61</u>	<u>81.45</u>
HMGE	<b>98.75</b>	<b>98.77</b>	<b>57.52</b>	<b>71.76</b>	<b>43.02</b>	<b>43.16</b>	<b>80.33</b>	<b>82.08</b>

TABLE 5: Ablation study on the task of node classification. Best in bold and second best underlined.

the link prediction and node classification results. The progressive refinement of the hidden relations allows to align a large number of divergent and complementary dimensions to a consensus embedding, which in turn alleviates the information loss caused by the widely used single and linear aggregation step. Note that the optimal number of layers depends on the dataset and should be selected through empirical tests.

4.4.2 Second Ablation ( $\alpha$  Weights)

We evaluate the performance of HMGE without using the  $\alpha$  weights. More precisely, we compare between using Eq. (6) to compute the non-linear combinations, and using the same formula without weights:

$$A_j^{(l)} = \sigma\left(\sum_{i=1}^{D_{l-1}} A_i^{(l-1)}\right). \tag{13}$$

The second rows of Tables 4 and 5 show the results obtained after performing the second ablation, respectively on the tasks of link prediction and node classification. We can see that the  $\alpha$  weights grant the trained model more flexibility to improve the link prediction and node classification results. These weights are necessary to adjust the importance of each latent dimension. Otherwise, all the dimensions would contribute equally to the generated latent structures. To identify relevant non-linear combinations,

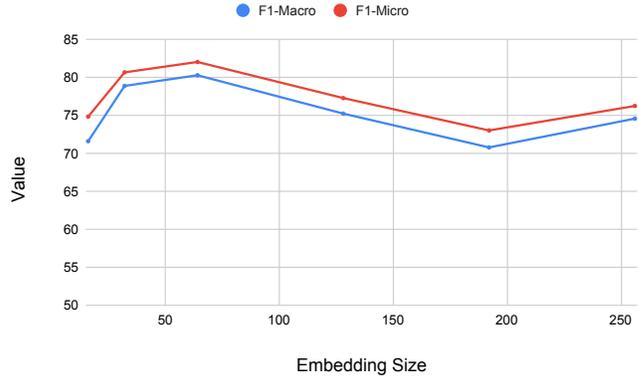


Fig. 4: Sensitivity to the embeddings size on STRING-DB

some dimensions must be attenuated or even discarded using the trainable  $\alpha$  weights.

4.5 Hyper-parameters Sensitivity

Fig. 4 shows the variations of F1-macro and F1-micro scores with respect to the node embedding size  $M$ . We can see that the variations remain small as the scores fluctuate within the range of 71% and 82%. Since HMGE yields consistent results for a wide range of values, we can conclude that our approach is robust with respect to the node embedding size.

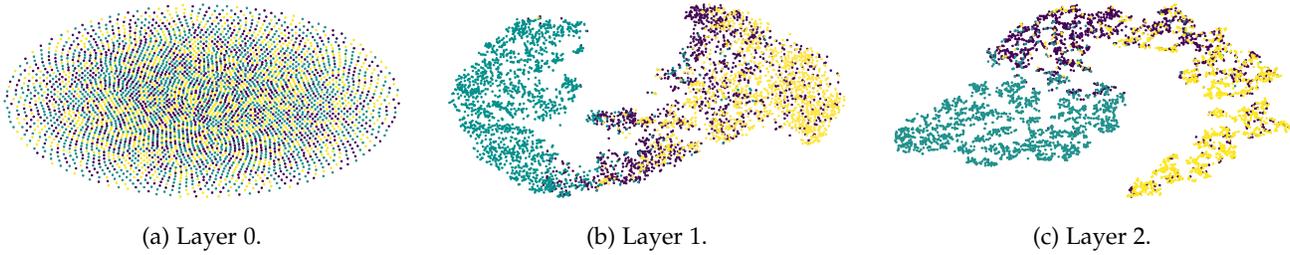


Fig. 5: Visualization of the embeddings learned at different layers of HMGE on STRING-DB.

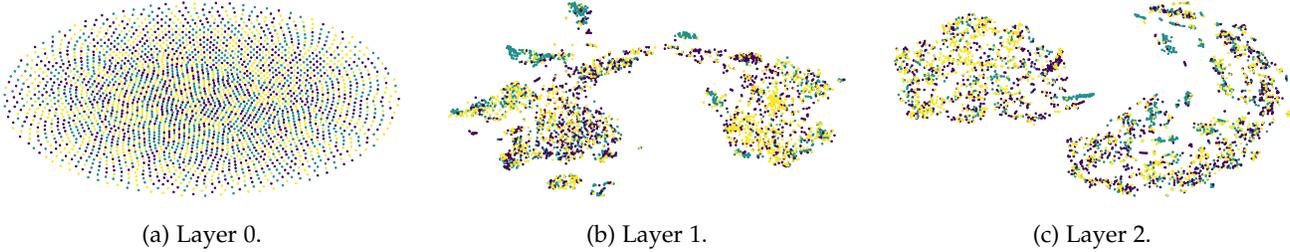


Fig. 6: Visualization of the embeddings learned at different layers of HMGE on IMDB.

## 4.6 Visualizations

In this subsection, we show qualitative results to give additional insights into the inner working of HMGE.

### 4.6.1 Node Embedding

Fig. 5 and 6 show 2D T-SNE visualizations of the embedding matrix  $H^{(l)}$  at multiple layers of HMGE, respectively on STRING-DB and IMDB. Different colors represent different classes. Generally, we can see that each layer improves upon the separation between the different classes. Note that layer 0 represents the input layer.

On STRING-DB, the classes cannot be linearly separated at layer 0. At layer 1, the blue class is well separated from the rest of the classes, but the purple and yellow classes are still tangled. Layer 2 improves the geometric configuration by pushing away samples from the yellow class from the purple one. We conclude that, for STRING-DB, one layer is sufficient to identify the blue class. A second layer is necessary to better separate the purple and yellow classes. This illustrates the benefits of a hierarchical aggregation approach to multiplex graph embedding. On IMDB, layer 1 forms the foundation for partitioning the nodes into 2 communities. Layer 2 improves on this separation. However, the classes are still not linearly separable. Since the model is trained without ground-truth labels, the representations may not align with all downstream tasks.

### 4.6.2 Combination Weights

Fig. 7 shows a visualization of the combination weights  $\alpha^{(1)}$  of the first layer on DBLP-Authors. The horizontal axis shows the different dimensions of the first layer. The vertical axis shows the weight of each dimension.

We can see that dimensions 3, 8, 9, and 10 have been assigned small weights compared to the other dimensions. In the ablation study, we showed that the combination weights have an important impact on the quality of the final embeddings. In particular, the results of link prediction on DBLP-Authors significantly drop when not using combination

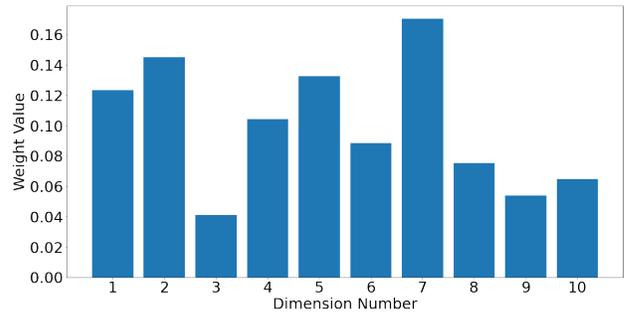


Fig. 7: Visualization of the combination weights  $\alpha^{(1)}$  on DBLP-Authors.

weights. The visual results confirm that some dimensions are less relevant than the others for link prediction.

## 5 CONCLUSION

In this paper, we propose HMGE, a novel approach for embedding high-dimensional multiplex graphs. In the high-dimensional setting, informative latent structures are hidden in non-linear combinations of the initial dimensions. In this context, traditional methods based on linear aggregation struggle to achieve suitable consensus embeddings. To tackle these issues, our approach hierarchically encodes the input multiplex graph to low-dimensional node representations using hierarchical aggregations. At each level of the hierarchy, hidden dimensions are formed by computing non-linear combinations of the input dimensions. This process gradually generates lower-dimensional multiplex graphs and identifies relevant latent structures hidden in the initial graph. Experiments on synthetic and real-world datasets show that our approach outperforms the state-of-the-art methods in multiplex graph embedding. The ablation study confirms the relevance and effectiveness of hier-

archical aggregations for high-dimensional multiplex graph embedding.

## ACKNOWLEDGMENTS

This work has been supported by Research Grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

## REFERENCES

- [1] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.
- [2] R. Oughtred, J. Rust, C. Chang, B.-J. Breitkreutz, C. Stark, A. Willems, L. Boucher, G. Leung, N. Kolas, F. Zhang *et al.*, "The biogrid database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions," *Protein Science*, vol. 30, no. 1, pp. 187–200, 2021.
- [3] X. Zhang, L. He, K. Chen, Y. Luo, J. Zhou, and F. Wang, "Multi-view graph convolutional network and its applications on neuroimaging analysis for parkinson's disease," in *AMIA Annual Symposium Proceedings*, vol. 2018. American Medical Informatics Association, 2018, pp. 1147–1156.
- [4] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi, "Multidimensional networks: foundations of structural analysis," *World Wide Web*, vol. 16, no. 5, pp. 567–593, 2013.
- [5] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, and X. He, "Multi-graph convolution collaborative filtering," in *International Conference on Data Mining*. IEEE, 2019, pp. 1306–1311.
- [6] R. J. Sánchez-García, E. Cozzo, and Y. Moreno, "Dimensionality reduction and spectral properties of multilayer networks," *Physical Review E*, vol. 89, no. 5, p. 052815, 2014.
- [7] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, "Structural reducibility of multilayer networks," *Nature Communications*, vol. 6, no. 1, pp. 1–9, 2015.
- [8] A. Solé-Ribalta, M. De Domenico, S. Gómez, and A. Arenas, "Random walk centrality in interconnected multilayer networks," *Physica D: Nonlinear Phenomena*, vol. 323, pp. 73–79, 2016.
- [9] M. El Gheche, G. Chierchia, and P. Frossard, "Orthonet: multilayer network data clustering," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 152–162, 2020.
- [10] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *The Web Conference*, 2020, pp. 3070–3076.
- [11] C. Park, D. Kim, J. Han, and H. Yu, "Unsupervised attributed multiplex network embedding," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5371–5378.
- [12] B. Jing, C. Park, and H. Tong, "Hdmi: High-order deep multiplex infomax," in *The Web Conference*, 2021, pp. 2414–2424.
- [13] X. Chu, X. Fan, D. Yao, Z. Zhu, J. Huang, and J. Bi, "Cross-network embedding for multi-network alignment," in *The Web Conference*, 2019, pp. 273–284.
- [14] L. Pio-Lopez, A. Valdeolivas, L. Tichit, É. Remy, and A. Baudot, "Multiverse: a multiplex and multiplex-heterogeneous network embedding approach," *Scientific Reports*, vol. 11, no. 1, pp. 1–20, 2021.
- [15] H. Zhang, L. Qiu, L. Yi, and Y. Song, "Scalable multiplex network embedding," in *International Joint Conferences on Artificial Intelligence*, vol. 18, 2018, pp. 3082–3088.
- [16] W. Liu, P.-Y. Chen, S. Yeung, T. Suzumura, and L. Chen, "Principled multilayer network embedding," in *International Conference on Data Mining Workshops*. IEEE, 2017, pp. 134–141.
- [17] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.
- [18] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Multi-task network embedding," *International Journal of Data Science and Analytics*, vol. 8, no. 2, pp. 183–198, 2019.
- [19] J. Ni, S. Chang, X. Liu, W. Cheng, H. Chen, D. Xu, and X. Zhang, "Co-regularized deep multi-network embedding," in *The Web Conference*, 2018, pp. 469–478.
- [20] R. Matsuno and T. Murata, "Mell: effective embedding method for multiplex networks," in *The Web Conference*, 2018, pp. 1261–1268.
- [21] Y. Ma, S. Wang, C. C. Aggarwal, D. Yin, and J. Tang, "Multi-dimensional graph convolutional networks," in *International Conference on Data Mining*. SIAM, 2019, pp. 657–665.
- [22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.
- [23] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2022.
- [24] O. Boutemine and M. Bouguessa, "Mining community structures in multidimensional networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 4, pp. 1–36, 2017.
- [25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [26] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.
- [27] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.
- [28] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.
- [29] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1358–1368.
- [30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, p. 3111–3119, 2013.
- [31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *The Web Conference*, 2015, pp. 1067–1077.
- [32] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.
- [33] A. Mitra, P. Vijayan, R. Sanasam, D. Goswami, S. Parthasarathy, and B. Ravindran, "Semi-supervised deep learning for multiplex networks," in *International Conference on Knowledge Discovery and Data Mining*, 2021, pp. 1234–1244.
- [34] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *International Conference on Learning Representations*, 2015.
- [35] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. Yu, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [36] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *International Conference on Learning Representations*, 2019.
- [37] J. Tang, "Aminer: Toward understanding big scholar data," in *International Conference on Web Search and Data Mining*, 2016, pp. 467–467.
- [38] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork *et al.*, "String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, no. 1, pp. 607–613, 2019.
- [39] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic block-models: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.